Getting Started in C

CSC 230 : C and Software Tools

NC State Department of Computer Science

Topics for Today

- C Overview
- Software Tools
- Building a program
- The common platform
- The C you already know

You **Want** to Learn C

- It’s fairly common, and shares a lot with many other, common languages
- It gives us a chance to think like ***procedural*** developers
- It’s a lower-level language than Java
  - Can offer much better performance (often not that important)
  - Exposes more of what’s going on underneath
  - this can make us more effective developers (even if we never program in C again)
- It will help prepare us for:
  - Operating Systems (CSC 246)
  - Assembly Language (CSC 236)

You Want to Learn C

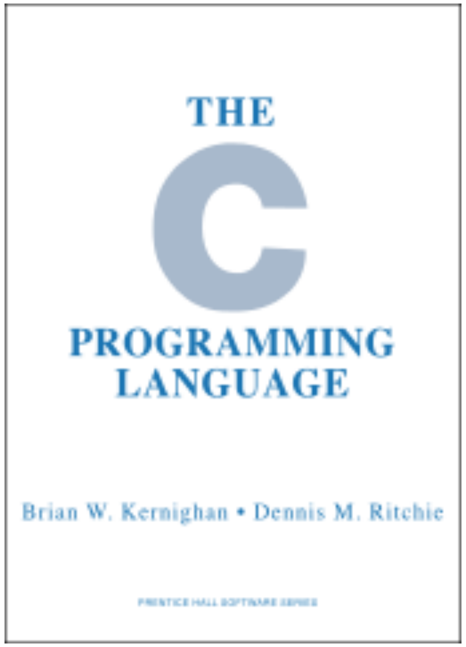
- Someone has to be able to program in C
  - That operating system you like to use, what do you think it's written in?
    - Linux : Assembly and C
    - MS Windows : Assembly, C and C++
  - How about that JVM that lets you run your fancy, high-level programs?
    - Assembly, C and C++
  - Lots of other examples
    - Embedded systems (cars, calculators, appliances, etc.)
    - High performance applications (science/engineering)

Thinking in C

- C : Programming in a different Type of language
  - Like Java, it's an *imperative language*, focused on **how** a computation should be performed
  - C is *procedural*
    - A program is a collection of procedures
    - Focus on **actions** performed by these procedures
  - Instead of *object-oriented*
    - A program is a collection of objects, each with state and operations
    - Focus on the **state** of these objects
- Of course, there are other ways we could go …
  - *Declarative Languages* : focus on **what** the program should compute rather than *how* it should compute it
    - *Functional Languages*: Lisp, Scheme, Haskell
    - *Dataflow Languages*
    - *Logic* or *Constraint-Based*: Prolog
    - *Markup Languages*: HTML

How We Got Here

- Developed along with the Unix operating system
  - An alternative to developing the OS in assembly
    - More portable
    - More readable/maintainable
- What's C
  - Informal standard for 10 years
  - C89 standard in 1989/1990
  - C99 standard in 1999


THE
C
PROGRAMMING LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES

C Strengths and Weaknesses

- Think of C as a thin veneer over the underlying assembly language
  - Lets us do some things we couldn't do in a higher level language ☺
  - Standard leaves some details implementation-defined, to better exploit the hardware ☺ and ☹
    - e.g., C has an int type, what's its range?
- C programs and the compiler can be tiny and exhibit a very small memory footprint ☺
  - Example trivial C and Java programs
- C offers very little in *protection* and *security* ☹
- C lacks some constructs for managing very large projects ☹

It’s Not Just About C

- Software tools
  - To help write, build, analyze and maintain software
  - Coordinate contributions from a team
- Examples:
  - Editors, pretty printers
  - Compilers, linkers
  - Debuggers
  - Code generators
  - Performance analyzers
- Often, these are integrated into the IDE
  - But, there’s some value in being able to run them directly

The Common Platform

- Different systems have different processors, line termination rules, compilers, language versions, etc.
- We need to agree on what system to target
- This is our *common platform*
  - Intel PC
  - Linux OS
  - gcc compiler suite
- Readily available on campus and from home

Choice in Where you Develop

Environment	GUI Interface?	Access to AFS Files?
Unity Computer Lab (e.g., EB3 2108)	Yes	Yes
ssh to remote-linux.eos.ncsu.edu	No Well, Yes with X11	Yes
ssh to VCL reservation	No Well, Yes with X11	Via sftp
Use Mac OS X with developer tools	Yes	Via sftp
Use MS Windows with cygwin	Yes	Via sftp or ExpandDrive
Use Linux on your PC or Mac (Dual boot or virtual machine)	Yes	Via sftp, etc.
Try our ready-made Centos6.6 VM image	Yes	Via sftp, ExpandDrive, etc.

Choice in Where you Develop

Environment	GUI Interface?	Access to AFS Files?
Unity Computer Lab (e.g., EB3 2108)	Yes	Yes
ssh to remote-linux.eos.ncsu.edu	No Well, Yes with X11	Yes
ssh to VCL reservation	No Well, Yes with X11	Via sftp
Use Mac OS X with developer tools	Yes	Via sftp
Use MS Windows with cygwin	Yes	Via sftp or ExpandDrive
Use Linux on your PC or Mac (Dual boot or virtual machine)	Yes	Via sftp, etc.
Use our ready-made Centos6.6 VM image	Yes	Via sftp, ExpandDrive, etc.

It's fine to develop here

Choice in Where you Develop

Environment	GUI Interface?	Access to AFS Files?
Unity Computer Lab (e.g., EB3 2108)	Yes	Yes
ssh to remote-linux.eos.ncsu.edu	No Well, Yes with X11	Yes
ssh to VCL reservation	No Well, Yes with X11	Via sftp
Use Mac OS X with developer tools	Yes	Via sftp
Use MS Windows with cygwin	Yes	Via sftp or ExpandDrive
Use Linux on your PC or Mac (Dual boot or virtual machine)	Yes	Via sftp, etc.
Use our ready-made Centos6.6 VM image	Yes	Via sftp, ExpandDrive, etc.

But, you should test here before you submit

Meet C

```
/**
  @file hello.c
  @author David Sturgill (dbsturgi)
  A program that prints: Hello World
 */
#include <stdio.h>

/**
  Starting point for the program.
  @return exit status
 */
int main()
{
  printf( "Hello World\n" );
  return 0;
}
```

Meet C

```c
/**
  @file hello.c
  @author David Sturgill (dbsturgi)
  A program that prints: Hello World
 */
#include <stdio.h>

/**
  Starting point for the program.
  @return exit status
 */
int main()
{
  printf( "Hello World\n" );
  return 0;
}
```

```
$ gcc -Wall -std=c99 hello.c -o hello
$ ./hello
```

Compile like this

Execute like this

What are You Looking At?

```
/**
  @file hello.c
  @author David Sturgill (dbsturgi)
  A program that prints: Hello World
 */
#include <stdio.h>

/**
  Starting point for the program.
  @return exit status
 */
int main()
{
  printf( "Hello World\n" );
  return 0;
}
```

A Comment, part of our style requirements

Telling the compiler about library components we use below (just printf).

A main function, where your program starts (see, it's not inside a class).

A call to the printf function to, well, print something out.

We're all done. Exit with success.

Building C Programs

- Here’s a recipe for building any simple C program:

Name of your source file

```
$ gcc -Wall -std=c99 X.c -o X
```

Enable lots of warnings.

Use the C99 Standard

Name of the resulting executable

The C You Already Know

- Some parts of the C language look a lot like Java

It all starts with main (but the parameters are different)

```
int main()
{
    int a = 25;
    double x = 3.14;
    char c = '*';

    .
    .
    .
}
```

You can declare and initialize local variables with types like **int**, **float**, **char** and **double**.

Before the C99 standard, local variables had to be declared at the top of a function or block.

The C You Already Know

- Expressions
  - You have a lot of the same operators (+, -, *, /, %, ++, --, etc.)

```
int a = 25;
double x = a * 1.5 – 18.7;

a++;
x = x / 2;


int *p = &a;
```

Looks just like good old Java.

Same here.

But, C has some operators Java doesn't have.

The C You Already Know

- Flow of control
  - You have an if statement that looks a lot like Java:

```
if ( a > 25 ) {
    a /= 2;
}
```

  - … and a for loop:

```
for ( int i = 0; i < 25; i++ ) {
    x += i;
}
```

The C You Already Know

- Flow of control
  - … and, there's a while loop:

```
    while ( x < 100.0 ) {
        x *= 1.05;
    }
```

  - … and a do/while, for the few times you need it.
  - … and switch
  - … and break and continue
  - … and one other thing …

The C You Already Know

- Functions
  - C has functions, with parameters and return types

```
double power( double x, int p )
{
    double result = 1;
    for ( int i = 0; i < p; i++ )
        result *= x;
    return result;
}
```

Here's a function definition.

```
        y = power( 3.25, j + 1 );
```

Here's a function call.

  - Like methods in Java.

The C You Already Know

- Some things are different.
  - C functions aren't part of a class, they are defined at *file scope*

```
double power( double x, int p )
{
    double result = 1;
    for ( int i = 0; i < p; i++ )
        result *= x;
    return result;
}
```

I'm not in a function, kind of like a static method in Java.

```
        y = power( 3.25, j + 1 );
```

And, C99 needs to see the function definition (or declaration) **before** you try to call it.

Not Quite Java

- **C doesn't force you to initialize your local variables.**
- **… and, it doesn't initialize them for you.**

```
double power( double x, int p )
{
    double result;
    for ( int i = 0; i < p; i++ )
        result *= x;
    return result;
}
```

Oops.
What value will this variable have?
Whatever was in that region of memory.

What will this return?
Probably garbage.

```
y = power( 3.25, j + 1 );
```